SENTENCIAS SQL

No se distingue mayúsculas ni minúsculas, ni espacios ni saltos de linea

COMANDOS DDL:

CREATE DATABASE nombrebasededatos;
create table tabla1 (columna1 varchar2 (25), columna2 (number (3));

create table nombre (columna1, varchar2 (2)

```
CREATE TABLE NOMBRETABLA (
   columna varchar2 (30) CONSTRAINT nom_UK UNIQUE,
   columna2 number (8) NOT NULL,
   fecha DEAFULT SYSDATE,
   nombre VARCHAR (8) DEFAULT ('raul')
   otroe VARCHAR (8) CONSTRAINT asd_as CHECK (otroe= 'raul'),
   numero NUMER (8) UNIQUE,
CONSTRAINT sdf_sdf CHECK(numero <30 AND numero between 10 and 15 and columna ='palabra1' OR columna='palabra2'));
```

Ejemplo: (Clave primaria compuesta)

```
CREATE TABLE USUARIOS (
        Nombre VARCHAR2 (25),
        Apellidos VARCHAR2 (30),
        F_Nacimiento DATE,
        CONSTRAINT Usu_PK PRIMARY KEY(Nombre, Apellidos, F_Nacimiento));
```

CLAVE FORÁNEA:
CONSTRAINT fk_constraint_name FOREIGN KEY (column_1, column2,…) (FOREIGN KEY se pone sólo si se especifica al final, después de la coma. Si no basta con REFERENCES)
REFERENCES parent_table_name(column1,column2,..)

Nombres restricciones: tab_col_re
PK, FK, NN, UK,CK

DROP TABLE NombreTabla [CASCADE CONSTRAINTS]; (Se borra donde sea FK también)
TRUNCATE TABLE - Borra datos pero no estructura
RENAME tablavieja TO nombrenuevo;

ALTER TABLE USUARIO RENAME COLUMN User TO Login;
ALTER TABLE USUARIO [RENAME , DROP, DISABLE, ENABLE] CONSTRAINT restriccion;
ALTER TABLE NombreTabla ADD [DROP, MODIFY] (Columna Tipo_Datos [Propiedades]
CREATE INDEX nombreindice ON nombretable (columnas);
CREATE USER NombreUsuario
IDENTIFIED BY ClaveAcceso
DEFAULT TABLESPACE tablespace
TEMPORARY TABLESPACE tablespace
[QUOTA int {K | M} ON tablespace
[QUOTA UNLIMITED ON tablespace]
[PROFILE perfil];
ALTER TABLE USUARIO ADD CONSTRAINT no_bre CHECK (length (columna)>5);

GRANT [REVOKE] SELECT, INSERT, UPDATE, DELETE ON T_PEDIDOS TO miusuario [WITH GRANT OPTION];

- **OPERADORES:**
  Permiten crear expresiones complejas. Pueden ser aritméticos (+, -, *, /, ...) o lógicos (< , >, , < >, And, Or, …).

  Operadores lógicos

Operadores:	Descripción:
AND	Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Devuelve el valor contrario de la expresión.

  Operadores de comparación

Operadores:	Descripción:
<	Menor que.
>	Mayor que.
< >	Distinto de.
< =	Menor o igual.
> =	Mayor o igual.
=	Igual.
BETWEEN	Se utiliza para especificar un intervalo de valores.
LIKE	Se utiliza para comparar.
IN	Se utiliza para especificar filas de una base de datos.

- **FUNCIONES:**
  Para conseguir valores complejos. Por ejemplo, la función promedio para obtener la media de un salario. Existen muchas funciones, aquí tienes la descripción de algunas.

  Funciones de agregado:

Función:	Descripción:
AVG	Calcula el promedio de los valores de un campo determinado.
COUNT	Devuelve el número de filas de la selección.
SUM	Devuelve la suma de todos los valores de un campo determinado.
MAX	Devuelve el valor más alto de un campo determinado.
MIN	Devuelve el valor mínimo de un campo determinado.

- **LITERALES:**
  Les podemos llamar también constantes y serán valores concretos, como por ejemplo un número, una fecha, un conjunto de caracteres, etc.

Literales:	Descripción:
23/03/97	Literal fecha.
María	Literal caracteres.
5	Literal número.

- **COMANDOS:**

  Comandos DDL:

Comando:	Descripción:
CREATE	Se utiliza para crear nuevas tablas, campos e índices.
DROP	Se utiliza para eliminar tablas e índices.
ALTER	Se utiliza para modificar tablas.

  Comandos DML:

Comando:	Descripción:
SELECT	Se utiliza para consultar filas que satisfagan un criterio determinado.
INSERT	Se utiliza para cargar datos en una única operación.
UPDATE	Se utiliza para modificar valores de campos y filas específicos.
DELETE	Se utiliza para eliminar filas de una tabla.

  Comandos DCL:

Comando:	Descripción:
GRANT	Permite dar permisos a uno o varios usuarios o roles para realizar tareas determinadas.
REVOKE	Permite eliminar permisos que previamente se han concedido con GRANT.

- **CLÁUSULAS:**
  Llamadas también condiciones o criterios, son palabras especiales que permiten modificar el funcionamiento de un comando.

Cláusulas:	Descripción:
FROM	Se utiliza para especificar la tabla de la que se van a seleccionar las filas.
WHERE	Se utiliza para especificar las condiciones que deben reunir las filas que se van a seleccionar.
GROUP BY	Se utiliza para separar las filas seleccionadas en grupos específicos.
HAVING	Se utiliza para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Se utiliza para ordenar las filas seleccionadas de acuerdo a un orden específico.

CONSULTAS:

SELECT [ALL | DISTINCT] columna1 “se puede renombrar”, columna2, ... FROM tabla1, tabla2, ... WHERE condición1, condición2, ... ORDER BY ordenación;

Consultas con operadores:
SELECT nombre, precio, precio*1.16 **AS precio_con_iva FROM** articulos;
Para concatenar con un texto ||

select NOMBRE , APELLIDOS from EMPLEADOS where SALARIO > 1300 + 50

Consultas con condiciones:
SELECT nombre, apellido1,apellido2 **FROM** personas **WHERE** edad>60 **OR** edad<20;
SELECT tipo,modelo,precio **FROM** piezas **WHERE** precio **BETWEEN** 3 **AND** 8;
SELECT tipo,modelo,precio **FROM** piezas **WHERE** precio **IN** (3,5,8); o in (‘r’,’s’)

SELECT apellido1 **FROM** Personas
WHERE apellido1 **LIKE** ’S_nchez’; **% serie cualquiera** **_ carácter cualquiera**

SELECT nombre, apellido1, apellido2, f_n fecha_nacimiento **FROM** alumnos
ORDER BY fecha_nacimiento **DESC;**

SELECT nombre, apellido1, apellido2, fecha_nacimiento, id_clase**FROM** alumnos
ORDER BY 2,3,nombre; **Numero columna o nombre**

NUMÉRICAS:

```
SELECT ABS(-17) FROM DUAL; -- Resultado: 17 no tiene en cuenta + o -
SELECT CEIL(17.4) FROM DUAL; -- Resultado: 18
SELECT FLOOR(17.4) FROM DUAL; -- Resultado: 17
SELECT MOD(15, 2) FROM DUAL; --Resultado: 1 Calcula el resto de dividir
SELECT POWER(4, 5) FROM DUAL; -- Resultado: 1024 EXPONENTE
SELECT ROUND(12.5874, 2) FROM DUAL; -- Resultado: 12.59 REDONDEA CON DECIMALES INDICADOS
SELECT SQRT(25) FROM DUAL; --Resultado: 5 RAIZ CUADRADA
```

```
quita decimales
SELECT TRUNC(127.4567, 2) FROM DUAL; -- Resultado: 127.45
SELECT TRUNC(4572.5678, -2) FROM DUAL; -- Resultado: 4500
SELECT TRUNC(4572.5678, -1) FROM DUAL; -- Resultado: 4570
SELECT TRUNC(4572.5678) FROM DUAL; -- Resultado: 4572
```

```
SELECT SIGN(-23) FROM DUAL; – Resultado: -1  Si positivo 1, si 0 = 0
```

CARACTERES:

```
SELECT LOWER('En MInúsculAS') FROM DUAL; --Resultado: en minúsculas
SELECT UPPER('En MAyúsculAS') FROM DUAL; --Resultado: EN MAYÚSCULAS
```

```
SENTENCIAS UTILIZADAS PARA HACER BUSQUEDAS IGNORANDO MAYUSCULAS:
SELECT * FROM JUEGOS WHERE UPPER(NOMBRE)='AJEDREZ';
SELECT * FROM JUEGOS WHERE LOWER(NOMBRE)='ajedrez';
```

```
SELECT REPLACE('correo@gmail.es', 'es', 'com') FROM DUAL; --
Resultado: correo@gmail.com
```

SUBSTR(cad, m, n) Extrae de la cadena (cad) "n" caracteres desde posición "m"

```
SELECT SUBSTR('1234567', 3, 2) FROM DUAL; --Resultado: 34
```

```
SELECT LENGTH('hola') FROM DUAL; --Resultado: 4
SELECT NVL(correo, 'No tiene correo') FROM USUARIOS; MUESTRA VALORES NULOS
```

TRIM(cad), LTRIM(cad), RTRIM(cad) Elimina espacios.

FECHAS:

```
SELECT SYSDATE FROM DUAL; --Resultado: 15/08/20
SELECT SYSTIMESTAMP FROM DUAL; --Resultado: 15/08/20 11:40:41,969000 +02:00
SELECT ADD_MONTHS('27/07/11', 5) FROM DUAL; --Resultado: 27/12/11
SELECT MONTHS_BETWEEN('12/07/11','12/03/11') FROM DUAL; --Resultado: 4
SELECT LAST_DAY('27/07/11') FROM DUAL; --Resultado: 31/07/11
SELECT NEXT_DAY('31/12/11','LUNES') FROM DUAL; --Resultado: 02/01/12
SELECT SYSDATE - 5 FROM DUAL; -- Devuelve la fecha correspondiente a 5
días antes de la fecha actual
```

AGRUPACION

FUNCIONES DE COLUMNA;

SUM (expresión)	MIN (expresión)
AVG (expresión)	MAX (expresión)
STDEV (expresión)	COUNT (nbcolumna)
STDEVP (expresión)	COUNT (*)

FUNCIÓN ([ALL| DISTINCT] Expresión)

ALL-Todas las filas

DISTINCT- Descarta valores repetidos

SELECT (AVG(ventas) * 3) + SUM(cuota) FROM …

Si utilizamos **COUNT(*)**, calcularemos el total de filas, incluyendo aquellas que contienen valores **NULL**.

```
SELECT COUNT(nombre) FROM Usuarios;
SELECT COUNT(*) FROM Usuarios;
SELECT SUM( credito) FROM Usuarios;

SELECT MIN(credito) FROM Usuarios;
SELECT MAX (credito) FROM Usuarios;
```

GROUP BY se colocan las columnas por las que vamos a agrupar. En la cláusula HAVING se especifica la condición que han de cumplir los grupos para que se realice la consulta.

sintaxis:

```
SELECT columna1, columna2, ...
FROM tabla1, tabla2, ...
[WHERE condición1, condición2, …]
[GROUP BY columna1, columna2, …
[HAVING condición ]]
[ORDER BY ordenación];
```

1. **WHERE** que filtra las filas según las condiciones que pongamos.
2. **GROUP BY** que crea una tabla de grupos nueva.
3. **HAVING** filtra los grupos.
4. **ORDER BY** que ordena o clasifica la salida.

```
SELECT provincia, SUM(credito) FROM Usuarios
GROUP BY provincia
HAVING   UPPER(provincia)   =   'SEVILLA'   OR   UPPER(provincia)=
'BADAJOZ';
```

EMPAREJAMIENTO DE TABLAS DIFERENTES EN UNA CONSULTA

```
SELECT Nombre, Apellido1, Apellido2, Fecha_inicio, Fecha_fin
FROM EMPLEADOS, HISTORIAL_LABORAL
WHERE HISTORIAL_LABORAL.Empleado_DNI= EMPLEADOS.DNI;
```

Tablas externas, o sin emparejamiento

```
SELECT D.NOMBRE_DPTO, D.JEFE, E.NOMBRE, E.APELLIDO1, E.APELLIDO2
FROM DEPARTAMENTOS D, EMPLEADOS E
WHERE D.JEFE = E.DNI(+);
```

Añadinos (+) para aceptar valores nulos

JOIN= para unir tablas. detecta claves de union (NATURAL JOIN);

```
SELECT E.Nombre, E.Apellido1, E.Apellido2, H.Fecha_inicio,
H.Fecha_fin
FROM EMPLEADOS E JOIN HISTORIAL_LABORAL H ON (H.Empleado_DNI=
E.DNI);
```

```
SELECT E.Nombre, E.Apellido1, E.Apellido2, H.Fecha_inicio,
H.Fecha_fin
FROM EMPLEADOS E JOIN HISTORIAL_LABORAL H USING (H.Empleado_DNI);
```

UNION: une varias consultas

```
SELECT NombreCia, Ciudad FROM  PROVEEDORES WHERE Pais = 'Alemania'
UNION
SELECT NombreCia, Ciudad  FROM CLIENTES WHERE Pais = ‘Alemania';
```

```
INTERSECT: IGUAL PERO ELIMINA FILAS DUPLICADAS
SELECT nombre, domicilio FROM ingles INTERSECT
SELECT nombre, domicilio FROM frances INTERSECT
SELECT nombre, domicilio FROM portugues;
```

```
MINUS:DEVUELVE RESULTADOS DE  PRIMERA CONSULTA SI NO ESTA EN LA SEGUNDA
SELECT nombre, domicilio FROM INGLES
MINUS
```

```
SELECT nombre,domicilio FROM PORTUGUES;
SUBCONSULTAS:
SELECT listaExpr
                FROM tabla
                WHERE expresión_o_columna OPERADOR
                ( SELECT expresion_o_columna
                FROM tabla);
```

El **OPERADOR** puede ser >, <, >=, <=, !=, = o IN. Las subconsultas que se utilizan con estos operadores **devuelven un único valor**

- Cuando el resultado de la subconsulta es **más de una fila**, SQL utiliza **palabras reservadas** entre el operador y la consulta. Estas son:
  - **ANY**. Compara con cualquier fila de la consulta. La instrucción es válida si hay un registro en la subconsulta que permite que la comparación sea cierta.
  - **ALL**. Compara con todas las filas de la consulta. La instrucción resultará cierta si es cierta la comparación con todas las filas devueltas por la subconsulta.
  - **IN**. No utiliza comparador, lo que hace es comprobar si el valor se encuentra en el resultado de la subconsulta.
  - **NOT IN**. Comprueba si un valor no se encuentra en una subconsulta.

En la siguiente consulta obtenemos el empleado que menos cobra:

```
SELECT nombre, salario
FROM EMPLEADOS
WHERE salario <= ALL (SELECT salario FROM EMPLEADOS);
```

Esa misma consulta se podría haber realizado utilizando la función de agregado o colectiva MIN de la siguiente forma:

```
SELECT nombre, salario
FROM EMPLEADOS
WHERE salario =  (SELECT MIN(salario) FROM EMPLEADOS);
```

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater
than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
```

```
FROM OrderDetails;
```

Anexo I.- Elementos del lenguaje SQL.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores, funciones y literales . Todos estos elementos se combinan en las instrucciones y se utilizan para crear, actualizar y manipular bases de datos.

COMANDOS:
Comandos DDL. Lenguaje de Definición de Datos:

Comandos DDL. Lenguaje de Definición de Datos.

Comando:	Descripción:
CREATE	Se utiliza para crear nuevas tablas, campos e índices.
DROP	Se utiliza para eliminar tablas e índices.
ALTER	Se utiliza para modificar tablas.

Comandos DML. Languaje de Manipulación de Datos:

Comandos DML. Lenguaje de Manipulación de Datos.

Comando:	Descripción:
SELECT	Se utiliza para consultar filas que satisfagan un criterio determinado.
INSERT	Se utiliza para cargar datos en una única operación.
UPDATE	Se utiliza para modificar valores de campos y filas específicos.
DELETE	Se utiliza para eliminar filas de una tabla.

Comandos DCL. Lenguaje de Control de Datos:

Comandos DCL. Lenguaje de Control de Datos.

Comando:	Descripción:
GRANT	Permite dar permisos a uno o varios usuarios o roles para realizar tareas determinadas.
REVOKE	Permite eliminar permisos que previamente se han concedido con GRANT.

CLÁUSULAS:
Llamadas también condiciones o criterios, son palabras especiales que permiten modificar el funcionamiento de un comando.

Operadores lógicos

Operadores de comparación

Cláusulas

Cláusulas:	Descripción:
FROM	Se utiliza para especificar la tabla de la que se van a seleccionar las filas.
WHERE	Se utiliza para especificar las condiciones que deben reunir las filas que se van a seleccionar.
GROUP BY	Se utiliza para separar las filas seleccionadas en grupos específicos.
HAVING	Se utiliza para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Se utiliza para ordenar las filas seleccionadas de acuerdo a un orden específico.

OPERADORES:
Permiten crear expresiones complejas. Pueden ser aritméticos (+, -, *, /, ...) o lógicos (< , >, , < >, And, Or, ...).

Operadores lógicos.

Operadores:	Descripción:
AND	Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Devuelve el valor contrario de la expresión.

Operadores de comparación.

Operadores:	Descripción:
<	Menor que.
>	Mayor que.
<>	Distinto de.
<=	Menor o igual.

Operadores:

Descripción:

>= Mayor o igual.

= Igual.

BETWEEN Se utiliza para especificar un intervalo de valores.

LIKE Se utiliza para comparar.

IN Se utiliza para especificar filas de una base de datos.

FUNCIONES:
Para conseguir valores complejos. Por ejemplo, la función promedio para

obtener la media de un salario. Existen muchas funciones, aquí tienes la descripción de algunas.

Funciones de agregado:

Funciones de agregado.

Función:	Descripción:
AVG	Calcula el promedio de los valores de un campo determinado.
COUNT	Devuelve el número de filas de la selección.
SUM	Devuelve la suma de todos los valores de un campo determinado.
MAX	Devuelve el valor más alto de un campo determinado.
MIN	Devuelve el valor mínimo de un campo determinado.

LITERALES:
Les podemos llamar también constantes y serán valores concretos, como por ejemplo un número, una fecha, un conjunto de caracteres, etc.

Literales

Literales:	Descripción:
23/03/97	Literal fecha.
María	Literal caracteres.
5	Literal número.

SQL PLUS;

Ver usuario conectado: select user from DUAL;

Ver tablas: select table_name from user_tables order by table_name;